HyperGraphRAG: Retrieval-Augmented Generation via Hypergraph-Structured Knowledge Representation

Haoran Luo[1,2], **Haihong E**[1*], **Guanting Chen**[1], **Yandan Zheng**[2], **Xiaobao Wu**[2], **Yikai Guo**[3], **Qika Lin**[4], **Yu Feng**[5], **Zemin Kuang**[6], **Meina Song**[1], **Yifan Zhu**[1], **Luu Anh Tuan**[2]
[1]Beijing University of Posts and Telecommunications [2]Nanyang Technological University
[3]Beijing Institute of Computer Technology and Application [4]National University of Singapore
[5]China Mobile Research Institute [6]Beijing Anzhen Hospital, Capital Medical University
haoran.luo@ieee.org, ehaihong@bupt.edu.cn, anhtuan.luu@ntu.edu.sg

Abstract

Standard Retrieval-Augmented Generation (RAG) relies on chunk-based retrieval, whereas GraphRAG advances this approach by graph-based knowledge representation. However, existing graph-based RAG approaches are constrained by binary relations, as each edge in an ordinary graph connects only two entities, limiting their ability to represent the n-ary relations ($n \geq 2$) in real-world knowledge. In this work, we propose **HyperGraphRAG**, the first hypergraph-based RAG method that represents n-ary relational facts via hyperedges. HyperGraphRAG consists of a comprehensive pipeline, including knowledge hypergraph construction, retrieval, and generation. Experiments across medicine, agriculture, computer science, and law demonstrate that HyperGraphRAG outperforms both standard RAG and previous graph-based RAG methods in answer accuracy, retrieval efficiency, and generation quality. Our data and code are publicly available[1].

1 Introduction

Retrieval-Augmented Generation (RAG) [10, 6] has advanced knowledge-intensive tasks by integrating knowledge retrieval with large language models (LLMs) [17, 28], thereby enhancing factual awareness and generation accuracy. Standard RAG typically relies on chunk-based retrieval, segmenting documents into fixed-length text chunks retrieved via dense vector similarity, which overlooks the relationships between entities. Recently, GraphRAG [2] has emerged as a promising direction that structures knowledge as a graph to capture inter-entity relations, with the potential to improve retrieval efficiency and knowledge-driven generation [18].

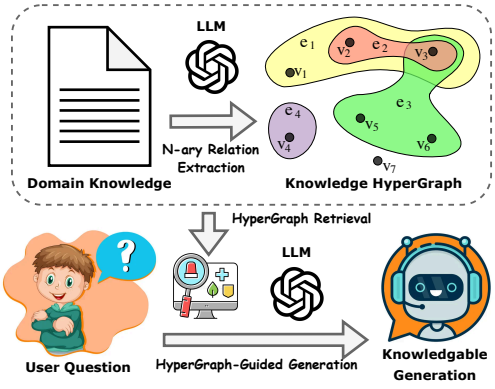


Figure 1: An illustration of HyperGraphRAG.

However, since each edge in an ordinary graph connects only two entities, existing graph-based RAG approaches [2, 7, 1, 8] are all restricted to **binary relations**, making them insufficient for modeling the **n-ary relations among more than two entities** that are widespread in real-world domain knowledge [25]. For example, in the medical domain, as illustrated in Figure 2, representing

* Corresponding author.
[1] https://github.com/LHRLAB/HyperGraphRAG

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

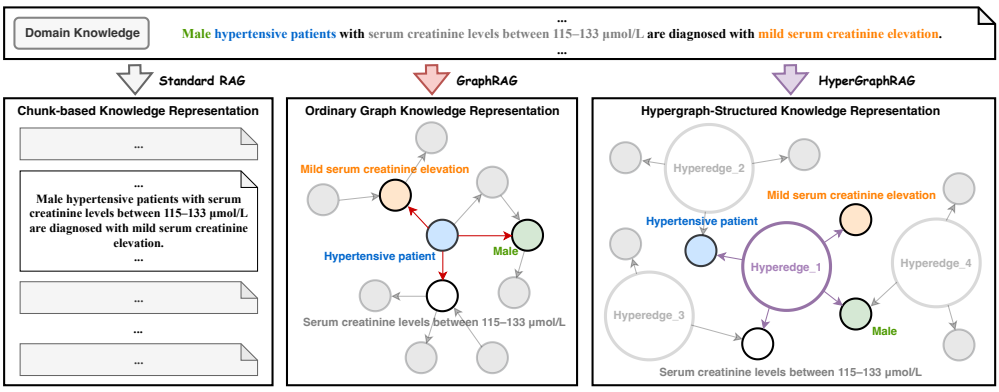


Figure 2: Comparison of knowledge representation: standard RAG uses chunks as units, GraphRAG captures binary relations with graphs, and HyperGraphRAG models n-ary relations with hyperedges.

the fact that "*Male hypertensive patients with serum creatinine levels between 115–133 μmol/L are diagnosed with mild serum creatinine elevation*" requires decomposing it into several binary relational triples, such as *Gender*:(*Hypertensive patient*, *Male*) and *Diagnosed_with*:(*Hypertensive patient*, *Mild serum creatinine elevation*), leading to representation sparsity during conversion process.

To address these limitations, we propose **HyperGraphRAG**, as illustrated in Figure 1, a novel graph-based RAG method built upon **hypergraph-structured knowledge representation**. In contrast to prior graph-based RAG methods constrained to binary relations, HyperGraphRAG leverages hyperedges to represent n-ary relational facts, where each hyperedge connects n entities ($n \geq 2$), e.g. *Hyperedge*:(*Hypertensive patient*, *Male*, *Serum creatinine levels between 115–133 μmol/L*, *Mild serum creatinine elevation*), and each hyperedge is expressed through natural language descriptions. This design ensures knowledge completeness, structural expressiveness, and inferential capability, thereby providing more comprehensive support for knowledge-intensive applications.

Our proposed HyperGraphRAG is built upon three key steps. First, we propose a **knowledge hypergraph construction method**, leveraging LLM-based n-ary relation extraction to extract and structure multi-entity relationships. The resulting hypergraph is stored in a bipartite graph database, with separate vector databases for entities and hyperedges to facilitate efficient retrieval. Second, we develop a **hypergraph retrieval strategy** that employs vector similarity search to retrieve relevant entities and hyperedges, ensuring that the knowledge retrieved is both precise and contextually relevant. Lastly, we introduce a **hypergraph-guided generation mechanism**, which combines retrieved n-ary facts with traditional chunk-based RAG passages, thereby improving response quality.

To validate the effectiveness, we conduct experiments in multiple knowledge-intensive domains [7], including medicine, agriculture, computer science, and law. Results demonstrate that HyperGraphRAG outperforms standard RAG and previous graph-based RAG methods in **answer accuracy**, **retrieval efficiency**, and **generation quality**, showcasing its strong potential for real-world applications.

2 Related Work

Graph-based RAG. GraphRAG [2] is the first graph-based RAG method that improves LLM generation via graph-based retrieval. Based on GraphRAG, several methods [26, 22, 11, 4, 23] focus on building graph-based RAG for different applications. LightRAG [7] enhances efficiency via graph indexing and updates. PathRAG [1] and HippoRAG2 [8] refine retrieval with path pruning and Personalized PageRank. However, all rely on binary relations, limiting knowledge expressiveness. In this work, we propose HyperGraphRAG, the first graph-based RAG method via hypergraph-structured knowledge representation. We compare several existing methods with HyperGraphRAG in Table 1.

Hypergraph Representation. Hypergraph-structured knowledge representation aims to overcome ordinary graph's limitations in modeling n-ary relations [15]. Early methods [25, 27, 12, 21] employ various embedding techniques to represent n-ary relational entities. Later methods [5, 24, 14] utilize GNN or attention to enhance embedding. However, existing methods mainly focus on link prediction, while hypergraphs also show potential for enhancing knowledge representation in graph-based RAG.

Table 1: Comparison of knowledge construction and retrieval methods for NaiveGeneration, StandardRAG, partial GraphRAG baselines, and our proposed HyperGraphRAG, where $\mathcal{K}$ represents the overall constructed knowledge, and K_q^* represents retrieved knowledge when given a user question q.

Method	Knowledge Construction	Knowledge Retrieval
NaiveGeneration	$\mathcal{K} = \emptyset$.	$K_q^* = \emptyset$.
StandardRAG	$\mathcal{K} = \{c_i\}_{i=1}^N$, where c_i is a chunk.	$K_q^* = K_{\text{chunk}} = \text{Top}_k\{\, c \in \mathcal{K} \mid \text{sim}(h_q, h_c)\,\}$
GraphRAG [2]	$\mathcal{K} = S = \{s_g \mid g \in \text{Community}(G)\}$, where S is the community summary set.	$K_q^* = \text{Detect}\{s_g \in S \mid q\}$, where detected community summaries are retrieved.
LightRAG [7]	$\mathcal{K} = G = (V, E)$, where V & E are entity & relation sets.	$K_q^* = \mathcal{F}\{v \in V, e \in E \mid q\} \cup K_{\text{chunk}}$, where entities & relations are retrieved with chunks.
PathRAG [1]	$\mathcal{K} = G = (V, E)$, where G is the same as LightRAG's.	$K_q^* = \text{Prune}\{p \in P_q \mid q\}$, where relational paths are retrieved via pruning.
HippoRAG2 [8]	$\mathcal{K} = G = (V \cup M,\ E)$, where V & M are phrase & passage nodes.	$K_q^* = \text{PageRank}\{m \in M \mid q\}$, where passages are retrieved via Personalized PageRank.
HyperGraphRAG (ours)	$\mathcal{K} = G_H = (V, E_H)$, where G_H is structured as a hypergraph.	$K_q^* = \mathcal{F}_n\{v \in V \mid q\} \cup \mathcal{F}_n\{e \in E_H \mid q\} \cup K_{\text{chunk}}$, where n-ary relational facts are retrieved with chunks.

3 Preliminaries

Definition 1: RAG. Given a question q and domain knowledge K, standard RAG first selects relevant document fragments d from K based on q, and then generates an answer y based on q and d. The probability model is formulated as:

$$P(y|q) = \sum_{d \in K} P(y|q,d)P(d|q,K). \tag{1}$$

Definition 2: Graph-based RAG. Graph-based RAG optimizes retrieval by representing knowledge as a graph structure $G = (V, E)$, where V is the set of entities and E is the set of relationships between entities. G consists of facts represented as $F = (e, V_e) \in G$, where e is the relation and V_e is the entity set connected to e. Given a question q, the retrieval process is defined as:

$$P(y|q) = \sum_{F \in G} P(y|q,F)P(F|q,G). \tag{2}$$

Definition 3: Hypergraph. A hypergraph $G_H = (V, E_H)$ [29] is a generalized graph, where V is the entity set, E_H is the hyperedge set, and each hyperedge $e_H \in E_H$ connects 2 or more entities:

$$V_{e_H} = (v_1, v_2, ..., v_n), \quad n \geq 2. \tag{3}$$

Unlike ordinary graphs, where relationships are binary $V_e = (v_h, v_t)$, hypergraphs model n-ary relational facts $F_n = (e_H, V_{e_H}) \in G_H$.

4 Method: HyperGraphRAG

In this section, we introduce the proposed HyperGraphRAG, as shown in Figure 3, including knowledge hypergraph construction, hypergraph retrieval strategy, and hypergraph-guided generation.

4.1 Knowledge Hypergraph Construction

To represent and store knowledge, we propose a knowledge hypergraph construction method that includes n-ary relational extraction, bipartite hypergraph storage, and vector representation storage.

N-ary Relation Extraction. To construct the knowledge hypergraph G_H, our first step is to extract multiple n-ary relational facts F_n from natural language documents $d \in K$. Unlike traditional hyper-relations [21], events [13], or other n-ary relation models [15], in the era of LLMs, to preserve richer and more diverse n-ary relations among entities, we propose a new n-ary relation representation $F_n = (e_H, V_{e_H})$, utilizing **natural language descriptions**, instead of structured relations, to represent hyperedges e_H among multiple entities V_{e_H} as follows.

(a) **Hyperedge:** Given an input text d, it is parsed into several independent knowledge fragments, each treated as a hyperedge: $E_H^d = \{e_1, e_2, ..., e_k\}$. Each hyperedge $e_i = (e_i^{\text{text}}, e_i^{\text{score}})$ consists of two parts: a natural language description e_i^{text}, and a confidence score $e_i^{\text{score}} \in (0, 10]$ indicating the association degree between e_i and d.

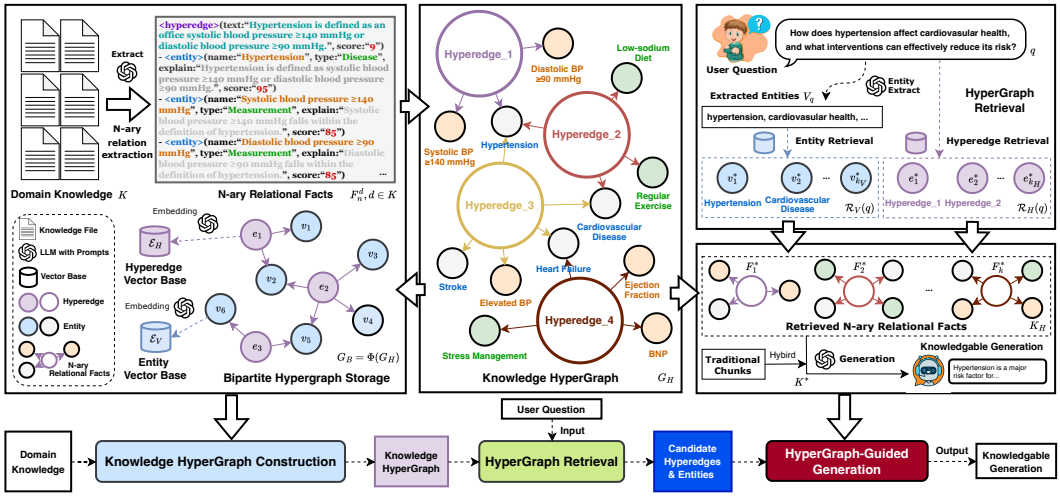


Figure 3: An overview of HyperGraphRAG, which constructs a knowledge hypergraph from domain knowledge, retrieves n-ary facts based on user questions, and generates knowledgeable responses.

(b) **Entity:** For each hyperedge e_i, entity recognition is performed to extract all contained entities: $V_{e_i} = \{v_1, v_2, ..., v_n\}$, where V_{e_i} is the entity set associated with e_i. Each entity $v_j = (v_j^{\text{name}}, v_j^{\text{type}}, v_j^{\text{explain}}, v_j^{\text{score}})$ consists of four parts: entity name $v_j^{\text{name}} \subseteq e_i^{\text{text}}$, type v_j^{type}, explanation v_j^{explain}, and confidence score $v_j^{\text{score}} \in (0, 100]$ indicating the extraction certainty.

Following this hypergraph-structured knowledge representation, we design an n-ary relation extraction prompt p_{ext}, detailed in Appendix A.1, to enable the LLM π to perform end-to-end knowledge fragment segmentation and entity recognition, thereby forming the n-ary relational fact set F_n^d:

$$F_n^d = \{f_1, f_2, ..., f_k\} \sim \pi(F_n | p_{\text{ext}}, d), \tag{4}$$

where each extracted n-ary relational fact $f_i = (e_i, V_{e_i})$ contains information about the corresponding hyperedge e_i and its associated entity set V_{e_i}. We convert all documents $d \in K$ into hyperedges and entities using n-ary relation extraction, forming a complete knowledge hypergraph G_H.

Proposition 1. *Hypergraph-structured knowledge representation is more comprehensive than binary.*

Proof. We provide experimental results in Section 5.4 and proofs in Appendix B.1. □

Bipartite Hypergraph Storage. After n-ary relation extraction, we store the constructed knowledge hypergraph G_H in a graph database to support an efficient query. We adopt an ordinary graph database represented as a bipartite graph structure $G_B = (V_B, E_B) = \Phi(G_H)$, to store the knowledge hypergraph $G_H = (V, E_H)$, where Φ is a transformation function defined as:

$$\Phi: \; V_B = V \cup E_H, \;\; E_B = \{(e_H, v) \mid e_H \in E_H, v \in V_{e_H}\}, \tag{5}$$

where V_B is the set of nodes in G_B, formed by merging the entity set V and the hyperedge set E_H from G_H. The edge set E_B captures the connections between each hyperedge $e_H \in E_H$ and its associated entities $v \in V_{e_H}$.

Based on G_B, we can efficiently query all entities associated with a hyperedge e_H or query all hyperedges linked to a specific entity v, thereby benefiting the optimized query efficiency of an ordinary graph database, as well as preserving the complete hypergraph-structured knowledge representation.

Moreover, G_B allows incremental updates through dynamically expansion: $G_B \leftarrow G_B \cup \Phi(G'_H)$, where G'_H represents newly added hypergraph information. The transformation of hyperedges and entities into the bipartite graph storage format enables seamless updates to the graph database.

Proposition 2. *A bipartite graph can losslessly preserve and query a knowledge hypergraph.*

Proof. We provide proofs in Appendix B.2. □

Vector Representation Storage. To support efficient semantic retrieval, we embed hyperedges $e_H \in E_H$ and entities $v \in V$ using the same embedding model f, ensuring that the vector representation of hyperedges and entities is in the same vector space as questions. Let Ψ be the vector function, then the vector representation storage for the knowledge hypergraph G_H is defined as: $\Psi(G_H) = (\mathcal{E}_H, \mathcal{E}_V)$, where $\mathcal{E}_H$ is the vector base of hyperedges and $\mathcal{E}_V$ is the vector base of entities:

$$\Psi: \ \mathcal{E}_H = \{\mathbf{h}_{e_H} \mid e_H \in E_H\}, \ \ \mathcal{E}_V = \{\mathbf{h}_v \mid v \in V\}, \tag{6}$$

where each hyperedge e_H and entity v in G_H is embedded into their vector representations: $\mathbf{h}_{e_H} = f(e_H)$, and $\mathbf{h}_v = f(v)$, respectively.

4.2 Hypergraph Retrieval Strategy

After constructing and storing the hypergraph G_H, we design an efficient retrieval strategy to match user questions with relevant hyperedges and entities.

Entity Retrieval. First, we extract key entities from the question q to facilitate subsequent matching. We design an entity extraction prompt $p_{\text{q\_ext}}$, detailed in Appendix A.2, along with the LLM π to extract the entity set V_q:

$$V_q \sim \pi(V|p_{\text{q\_ext}}, q). \tag{7}$$

After extracting entities, we retrieve the most relevant entities from the entity set V of the knowledge hypergraph G_H. We define the entity retrieval function $\mathcal{R}_V$, which retrieves the most relevant entities from $\mathcal{E}_V$ using cosine similarity:

$$\mathcal{R}_V(q) = \underset{v \in V}{\operatorname{argmax}}^{k_V} \left(\text{sim}(\mathbf{h}_{V_q}, \mathbf{h}_v) \odot v^{\text{score}}\right)_{>\tau_V}, \tag{8}$$

where $\mathbf{h}_{V_q} = f(V_q)$ is the concatenated text vector representation of the extracted entity set V_q, $\mathbf{h}_v \in \mathcal{E}_V$ is the vector representation of entity v, $\text{sim}(\cdot,\cdot)$ denotes the similarity function, $\odot$ represents element-wise multiplication between similarity and entity relevance score v^{score} determining the final ranking score, τ_V is the threshold for the entity retrieval score, and k_V is the limit on the number of retrieved entities.

Hyperedge Retrieval. Moreover, to expand the retrieval scope and capture complete n-ary relations within the hyperedge set E_H of the knowledge hypergraph G_H, we define the hyperedge retrieval function $\mathcal{R}_H$, which retrieves a set of hyperedges related to q:

$$\mathcal{R}_H(q) = \underset{e_H \in E_B}{\operatorname{argmax}}^{k_H} \left(\text{sim}(\mathbf{h}_q, \mathbf{h}_{e_H}) \odot e_H^{\text{score}}\right)_{>\tau_H}, \tag{9}$$

where $\mathbf{h}_q = f(q)$ is the text vector representation of q, $\mathbf{h}_{e_H} \in \mathcal{E}_H$ is the vector representation of the hyperedge e_H, $\odot$ represents element-wise multiplication between similarity and hyperedge relevance score e_H^{score} determining the final ranking score, τ_H is the threshold for the hyperedge retrieval score, and k_H limits the number of retrieved hyperedges.

4.3 Hypergraph-Guided Generation

To fully utilize the structured knowledge in the hypergraph, we propose a Hypergraph-Guided Generation mechanism, which consists of hypergraph knowledge fusion and generation augmentation.

Hypergraph Knowledge Fusion. The primary goal of hypergraph knowledge fusion is to expand and reorganize the retrieved n-ary relational knowledge to form a comprehensive knowledge input. Since q may only match partial entities or hyperedges, we further expand the retrieval scope. To obtain a complete set of n-ary relational facts, we design a bidirectional expansion strategy, that includes expanding hyperedges from retrieved entities and expanding entities from retrieved hyperedges.

First, given the entity set retrieved from q, denoted as $\mathcal{R}_V(q) = \{v_1, v_2, ..., v_{k_V}\}$, we retrieve all hyperedges in the knowledge hypergraph G_H that connect these entities:

$$\mathcal{F}_V^* = \bigcup_{v_i \in \mathcal{R}_V(q)} \{(e_H, V_{e_H}) \mid v_i \in V_{e_H}, e_H \in E_H\}. \tag{10}$$

Next, we expand the set of entities connected to the retrieved hyperedges $\mathcal{R}_H(q) = \{e_1, e_2, ..., e_{k_H}\}$:

$$\mathcal{F}_H^* = \bigcup_{e_i \in \mathcal{R}_H(q)} \{(e_i, V_{e_i}) \mid V_{e_i} \subseteq V\} \tag{11}$$

Finally, we merge the expanded hyperedge set $\mathcal{F}_V^*$ with the expanded entity set $\mathcal{F}_H^*$ to form a complete retrieved n-ary relational fact set $K_H = \mathcal{F}_V^* \cup \mathcal{F}_H^*$. This set contains all necessary n-ary relational knowledge for reasoning and generation, ensuring a comprehensive input for the LLM.

Generation Augmentation. Following hypergraph knowledge fusion, we augment the generation strategy to improve the accuracy and readability of the responses. We adopt a hybrid RAG fusion mechanism, combining hypergraph knowledge K_H with retrieved chunk-based text fragments K_{chunk} to form the final knowledge input. We define the final knowledge input $K^* = K_H \cup K_{\text{chunk}}$, where K_{chunk} consists of chunk-based text fragments retrieved using traditional RAG.

Finally, we use a retrieval-augmented generation prompt p_{gen}, detailed in Appendix A.3, that combines hypergraph knowledge K^* and the user question q as input to LLM π to generate final response y^*:

$$y^* \sim \pi(y|p_{\text{gen}}, K^*, q). \tag{12}$$

Proposition 3. *Retrieving knowledge on a knowledge hypergraph improves retrieval efficiency compared to methods based on ordinary binary graphs, leading to gains in generation quality.*

Proof. We provide experimental results in Sections 5.5 and 5.6 and proofs in Appendix B.3. □

5 Experiments

This section presents the experimental setup, main results, and analysis. We answer the following research questions (RQs): **RQ1:** Does HyperGraphRAG outperform other methods? **RQ2:** Does the main component of HyperGraphRAG work? **RQ3:** How effective is the knowledge hypergraph constructed by HyperGraphRAG across various domains? **RQ4:** Could the hypergraph retrieval strategy improve retrieval efficiency? **RQ5:** How effective is the generation quality of HyperGraphRAG? **RQ6:** How are the time and cost of HyperGraphRAG in construction and generation phases?

5.1 Experimental Setup

Datasets. To evaluate the performance of HyperGraphRAG across multiple domains, we select four knowledge contexts from UltraDomain [19], as used in LightRAG [7]: **Agriculture**, Computer Science (**CS**), **Legal**, and a mixed domain (**Mix**). In addition, we include the latest international hypertension guidelines [16] as the foundational knowledge for the **Medicine** domain. For each of the five domains, we sample knowledge fragments one, two, and three hops away to construct questions with ground-truth answers verified by human annotators. We then categorize the questions into **Binary Source** and **N-ary Source**, based on whether the sampled knowledge of the question contains facts among n entities ($n > 2$). More details can be found in Appendix D.

Baselines. We compare HyperGraphRAG against six publicly available baseline methods: **Naive-Generation** [17], which directly generates responses using LLM; **StandardRAG** [6], a traditional chunk-based RAG approach; **GraphRAG** [2], **LightRAG** [7], **PathRAG** [1], and **HippoRAG2** [8], which are four selected available graph-based RAG methods described in Table 1. To ensure fairness, we use the same generation prompt, which can be found in Appendix E.

Evaluation Metrics. We evaluate the answer accuracy, retrieval efficiency, and generation quality of HyperGraphRAG and its baselines using 3 key metrics: **F1**, Retrieval Similarity (**R-S**), and Generation Evaluation (**G-E**). F1 measures word-level similarity between the generated answer and the ground-truth answer, following FlashRAG [9]. R-S assesses the semantic similarity between the retrieved knowledge and the ground-truth knowledge used to construct the question, in line with RAGAS [3]. G-E, inspired by HelloBench [20], is a metric that uses LLM-as-a-judge to evaluate generation quality in 7 dimensions and reports the average score. Details are provided in Appendix E.

Implementation Details. We use OpenAI's `GPT-4o-mini` for extraction and generation, and `text-embedding-3-small` for vector. During retrieval, we set the following parameters: entity retrieval $k_V = 60$, $\tau_V = 50$; hyperedge retrieval $k_H = 60$, $\tau_H = 5$; and chunk retrieval $k_C = 5$, $\tau_C = 0.5$. All experiments were conducted on a server with an 80-core CPU and 512GB RAM.

5.2 Main Results (RQ1)

To evaluate the effectiveness of HyperGraphRAG, we compare its performance with various baselines across multiple domains. The results are shown in Table 2.

Table 2: Performance comparison across different domains. **Bold** indicates the best performance.

Method	Medicine			Agriculture			CS			Legal			Mix		
	F1	R-S	G-E	F1	R-S	G-E	F1	R-S	G-E	F1	R-S	G-E	F1	R-S	G-E
Binary Source															
NaiveGeneration	12.63	0.00	44.70	11.71	0.00	45.76	18.93	0.00	48.79	22.91	0.00	50.00	18.58	0.00	46.14
StandardRAG	26.87	61.08	56.24	28.31	42.69	57.58	28.87	49.44	57.10	37.19	52.21	59.85	47.57	46.79	67.42
GraphRAG	17.13	54.56	48.19	20.67	40.90	52.41	23.75	37.65	53.17	31.09	34.26	54.62	23.62	25.01	48.12
LightRAG	12.16	52.38	44.15	17.70	41.24	50.32	22.59	41.86	51.62	33.63	45.54	56.42	29.98	34.22	54.50
PathRAG	14.74	52.30	45.36	21.97	42.21	53.13	25.28	41.49	53.28	32.32	43.60	55.45	40.87	33.36	60.75
HippoRAG2	21.12	57.50	51.08	12.60	16.85	44.56	16.94	21.05	47.29	20.10	34.13	46.77	21.10	18.34	45.83
HyperGraphRAG (ours)	**36.45**	**69.91**	**60.65**	**34.80**	**61.97**	**59.99**	**31.60**	**60.94**	**57.54**	**44.42**	**60.87**	**63.53**	**51.51**	**67.34**	**68.76**
N-ary Source															
NaiveGeneration	13.15	0.00	41.83	13.78	0.00	47.93	18.37	0.00	48.94	20.37	0.00	48.09	15.29	0.00	45.16
StandardRAG	28.93	64.06	55.08	26.55	48.93	56.62	28.99	47.35	56.69	37.50	51.16	60.09	38.83	47.73	61.82
GraphRAG	18.07	57.22	47.09	21.90	41.27	53.49	22.90	39.97	53.76	29.12	34.11	53.76	14.93	24.32	42.32
LightRAG	13.43	54.67	41.86	18.78	42.44	50.92	22.85	41.19	52.20	29.64	44.47	54.65	24.08	33.22	50.83
PathRAG	15.14	54.08	42.77	20.64	42.53	51.83	28.18	42.29	54.97	30.27	44.47	55.26	33.27	34.11	57.47
HippoRAG2	21.56	61.54	48.06	12.66	20.32	45.14	17.75	26.92	48.44	16.95	34.72	45.09	21.95	18.49	46.87
HyperGraphRAG (ours)	**34.26**	**70.48**	**58.06**	**32.98**	**62.58**	**59.59**	**31.00**	**59.25**	**58.35**	**43.20**	**60.07**	**63.70**	**45.91**	**69.09**	**65.04**
Overall															
NaiveGeneration	12.89	0.00	43.27	12.74	0.00	46.85	18.65	0.00	48.87	21.64	0.00	49.05	16.93	0.00	45.65
StandardRAG	27.90	62.57	55.66	27.43	45.81	57.10	28.93	48.40	56.89	37.34	51.68	59.97	43.20	47.26	64.62
GraphRAG	17.60	55.89	47.64	21.28	41.08	52.95	23.33	38.81	53.47	30.11	34.18	54.19	19.27	24.67	45.22
LightRAG	12.79	53.52	43.00	18.24	41.84	50.62	22.72	41.53	51.91	31.64	45.00	55.53	27.03	33.72	52.67
PathRAG	14.94	53.19	44.06	21.30	42.37	52.48	26.73	41.89	54.13	31.29	44.03	55.36	37.07	33.73	59.11
HippoRAG2	21.34	59.52	49.57	12.63	18.58	44.85	17.34	23.99	47.87	18.53	34.42	45.93	21.53	18.42	46.35
HyperGraphRAG (ours)	**35.35**	**70.19**	**59.35**	**33.89**	**62.27**	**59.79**	**31.30**	**60.09**	**57.94**	**43.81**	**60.47**	**63.61**	**48.71**	**68.21**	**66.90**

Overall Comparison Across Methods. HyperGraphRAG consistently outperforms all baselines across F1, R-S, and G-E metrics. Compared to StandardRAG, it achieves gains of +7.45 (F1), +7.62 (R-S), and +3.69 (G-E). Interestingly, existing graph-based RAG baselines often underperform StandardRAG, as their reliance on binary relational graphs causes knowledge fragmentation, sparsified retrieval, and incomplete context reconstruction during generation.

Comparison Across Source Types. HyperGraphRAG maintains strong gains under both Binary and N-ary settings. For Binary Source, it improves F1, R-S, and G-E by +8.6, +8.8, and +4.4; for N-ary Source, the improvements are +5.3, +6.4, and +2.9, confirming its robustness.

Comparison Across Domains. Performance gains are consistent across domains, with the largest improvements in Medicine and Legal (over +7 F1), and stable advantages in Agriculture and CS. HyperGraphRAG adapts well to both highly structured and more general knowledge tasks.

5.3 Ablation Study (RQ2)

As shown in Figure 4, we conduct an ablation study in the Medicine domain by removing entity retrieval (w/o ER), hyperedge retrieval (w/o HR), and their combination (w/o ER & HR). We also remove chunk retrieval fusion (w/o CR), and all modules (w/o ER & HR & CR):

Impact of Entity Retrieval (ER). ER is critical for precise retrieval by anchoring key concepts. Without ER, F1 falls from 35.4 to 29.8, underscoring its importance in selecting relevant entities for accurate generation.

Impact of Hyperedge Retrieval (HR). HR captures n-ary, multi-entity facts necessary for complex reasoning. Removing HR drops F1 from 35.4 to 26.4, highlighting its unique role beyond mere entity retrieval.

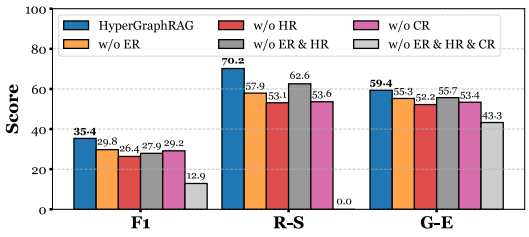
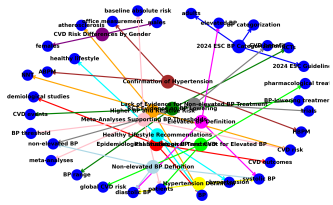


Figure 4: Results of the ablation study.

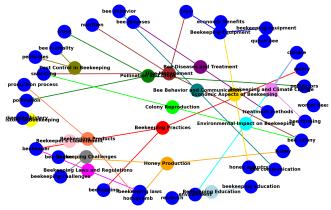
Impact of Chunk Retrieval Fusion (CR). CR enhances retrieval by integrating unstructured text with hypergraph data. Excluding CR reduces F1 from 35.4 to 29.2, demonstrating that the fusion leads to more complete and fluent generation.

5.4 Analysis of Hypergraph-structured Knowledge Representation (RQ3)

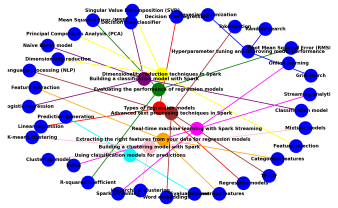
As shown in Figure 5, we assess HyperGraphRAG's knowledge representation across 5 domains:



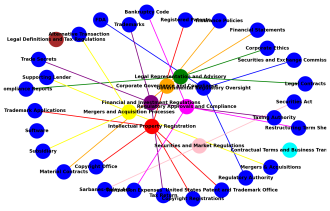
(a) Medicine HyperGraph



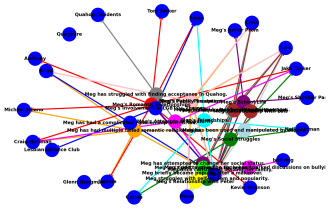
(b) Agriculture HyperGraph



(c) CS HyperGraph



(d) Legal HyperGraph



(e) Mix HyperGraph

	Med.	Agric.	CS	Legal	Mix
#Knowl. Token	179k	382k	795k	940k	122k
GraphRAG					
#Entity	329	699	1449	1711	225
#Community	256	523	930	517	59
LightRAG					
#Entity	3,725	5,032	8,967	5,354	2,229
#Relation	1,304	3,105	5,632	6,002	940
HyperGraphRAG					
#Entity	7,675	16,805	19,913	11,098	6,201
#Hyperedge	4,818	16,102	26,902	18,285	4,356

(f) Statistics of Construction

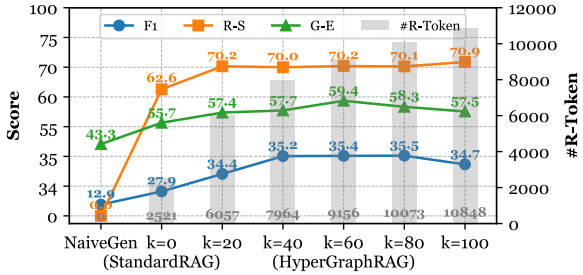
Figure 5: (a-e) Visualizations of knowledge hypergraphs constructed in 5 domains. (f) Statistical comparison highlights HyperGraphRAG's richer expressiveness over GraphRAG and LightRAG.

Visualization of Knowledge Structures. As shown in Figure 5(a)-5(e), unlike previous graph-based RAG methods, which only model binary relations, HyperGraphRAG connects multiple entities via hyperedges, forming a more interconnected and expressive network.

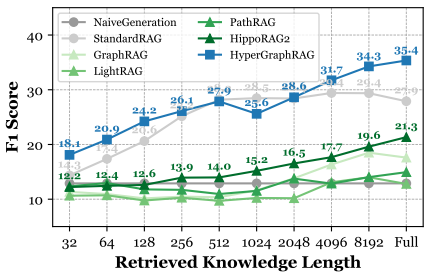
Statistical Analysis. As shown in Figure 5(f), HyperGraphRAG surpasses GraphRAG and LightRAG in all domains. For instance, in CS, it constructs 26,902 hyperedges, whereas GraphRAG has 930 communities and LightRAG 5,632 relations, showing a stronger capacity for capturing knowledge.

5.5 Analysis of Hypergraph Retrieval Efficiency (RQ4)

As shown in Figure 6, to evaluate retrieval efficiency, we conduct two experiments: (a) examining how HyperGraphRAG's retrieval efficiency and token length scales with different top-k values and (b) comparing its F1 scores with other methods under varying retrieval length limits:



(a) Impact of Top-k on Retreval Efficency & Token Length.



(b) F1 Comparison under Limited Lengths.

Figure 6: Experimental results in the Medicine domain analyzing hypergraph retrieval efficiency.

Impact of Retrieved Hyperedge Quantity. As shown in Figure 6(a), increasing the top-k hyperedges improves F1, R-S, and G-E, along with the rise in token count. Performance saturates around k = 60, indicating that HyperGraphRAG achieves strong retrieval quality with limited input.

Performance under Constrained Retrieval Length. As illustrated in Figure 6(b), HyperGraphRAG outperforms all binary graph-based RAG methods even under retrieval length limits, demonstrating the efficiency of n-ary representations and highlighting the semantic loss inherent in binary structures.

5.6 Analysis of Hypergraph-Guided Generation Quality (RQ5)

As shown in Figure 7, we evaluate the quality of the generation in seven dimensions:

Best Overall Generation Quality. HyperGraphRAG achieves the highest Overall score (61.5), significantly outperforming all baseline methods, indicating the comprehensive advantage in hypergraph-guided generation.

Lead on Key Dimensions. HyperGraphRAG achieves notable improvements in Correctness (64.8), Relevance (66.0), and Factuality (64.2), outperforming both standard RAG and binary graph-based methods. These gains indicate its strong capacity to produce accurate, context-aware, and knowledge-grounded responses.

Balanced Performance. Although the Diversity score (47.0) is relatively lower than other dimensions, HyperGraphRAG still exceeds all baselines, indicating that it maintains a balanced dimension-wise performance, effectively combining content richness with structural consistency for stable and high-quality generation.

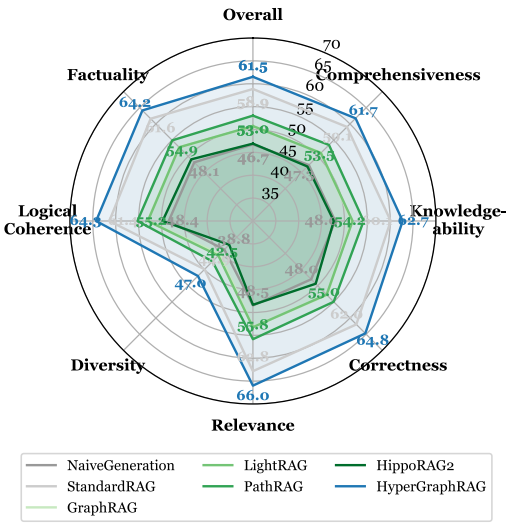


Figure 7: Generation Equality Evaluations.

5.7 Analysis of Time and Cost in Construction and Generation Phases (RQ6)

As shown in Table 3, to evaluate the efficiency and cost of HyperGraphRAG, we compare different methods in terms of knowledge construction and generation. We assess time consumption per 1k tokens (TP1kT), cost per 1k tokens (CP1kT), time per query (TPQ), and cost per 1k query (CP1kQ).

Time & Cost in Construction Phase. HyperGraphRAG demonstrates efficient knowledge construction with a time cost of 3.084 seconds per 1k tokens (TP1kT) and a monetary cost of $0.0063 per 1k tokens (CP1kT). This places it between the faster HippoRAG2 (2.758s, $0.0056) and slower GraphRAG (9.272s, $0.0058). While its cost is slightly higher than GraphRAG, HyperGraphRAG achieves a better balance between speed, expressiveness, and structure, offering a more compact yet richer representation of n-ary relational knowledge.

Table 3: Time & Cost Comparisons.

Method	Construction		Generation	
	TP1kT	CP1kT	TPQ	CP1kQ
NaiveGeneration	0 s	0 $	0.131 s	0.059 $
StandardRAG	0 s	0 $	0.147 s	1.016 $
GraphRAG	9.272 s	0.0058 $	0.221 s	1.836 $
LightRAG	5.168 s	0.0081 $	0.359 s	3.359 $
PathRAG	5.168 s	0.0081 $	0.436 s	3.496 $
HippoRAG2	2.758 s	0.0056 $	0.240 s	3.438 $
HyperGraphRAG	3.084 s	0.0063 $	0.256 s	3.184 $

Time & Cost in Generation Phase. During the generation phase, HyperGraphRAG requires 0.256 seconds per query (TPQ) and incurs a cost of $3.184 per 1k queries (CP1kQ). This is moderately higher than StandardRAG (0.147s, $1.016) but significantly lower than PathRAG (0.436s, $3.496) and LightRAG (0.359s, $3.359). Compared to GraphRAG (0.221s, $1.836), HyperGraphRAG slightly increases time and cost but compensates with better retrieval quality and generation outcomes. The results suggest that HyperGraphRAG achieves a favorable trade-off between generation efficiency and output quality, suitable for real-world knowledge-intensive applications.

6 Conclusion

In this work, we present HyperGraphRAG, a retrieval-augmented generation framework that models knowledge as hypergraphs to capture n-ary relational structures. By introducing novel methods for knowledge hypergraph construction, retrieval, and generation, HyperGraphRAG addresses limitations of binary graph-based RAG methods. Experimental results across diverse domains demonstrate consistent improvements in answer accuracy, retrieval relevance, and generation quality, confirming the effectiveness and generalizability of hypergraph-guided retrieval and generation.